Machine learning classification of new asteroid families members

V. Carruba[1⋆], S. Aljbaae[2], R. C. Domingos[3], A. Lucchini[1], P. Furlaneto[1].
[1] *São Paulo State University (UNESP), School of Natural Sciences and Engineering, Guaratinguetá, SP, 12516-410, Brazil*
[2] *National Space Research Institute (INPE), Division of Space Mechanics and Control, C.P. 515, 12227-310, São José dos Campos, SP, Brazil*
[3] *São Paulo State University (UNESP), Sao João da Boa Vista, SP, 13874-149, Brazil*

Accepted 2020 May 21. Received 2020 May 19 ; in original form 2020 March 20.

ABSTRACT
Asteroid families are groups of asteroids that are the product of collisions or of the rotational fission of a parent object. These groups are mainly identified in proper elements or frequencies domains. Because of robotic telescope surveys, the number of known asteroids has increased from $\simeq 10,000$ in the early 90's to more than $750,000$ nowadays. Traditional approaches for identifying new members of asteroid families, like the hierarchical clustering method (HCM), may struggle to keep up with the growing rate of new discoveries. Here we used machine learning classification algorithms to identify new family members based on the orbital distribution in proper $(a, e, \sin(i))$ of previously known family constituents. We compared the outcome of nine classification algorithms from stand alone and ensemble approaches. The Extremely Randomized Trees (ExtraTree) method had the highest precision, enabling to retrieve up to 97% of family members identified with standard HCM.

Key words: Minor planets, asteroids: general,–celestial mechanics, methods: data analysis.

1 INTRODUCTION

Asteroid families are groups of asteroids that form because of collisions or the rotational failure of a parent body. The most widely used method for identifying these groups is the hierarchical clustering method (HCM) in the $(a, e, \sin(i))$ proper element domain, where a is the proper semi-major axis, e is the proper eccentricity, and i the proper inclination. In this method, asteroids are linked to a parent body if their distance in proper element domains, defined through a metric, is less than a critical value, called cutoff. If an asteroid is closer to the parent body than this distance, it is added to the family list. The procedure is then repeated with this asteroid as a new parent body, until no new family members are identified.

This approach was introduced in the early 90's of the last century when the number of asteroids with proper elements was of the order of $\simeq 10,000$ (Zappalá et al. 1990, 1995). Thanks to robotic surveys such as Spacewatch and LINEAR, the number of asteroids for which reliable proper elements are available is nowadays of the order of 750,000, and this number is continually increasing (see DeMeo et al. (2015) for an in-depth discussion of the recent rates of asteroid discoveries). In high number density regions of the main belt, standard application of HCM may not be directly applicable: asteroid families close in proper element spaces may overlap and no longer be recognizable as an individual entity. While alternative implementations of HCM have been proposed to solve this problem (Milani et al. 2014), these approaches may be computationally expensive and require to obtain solutions for all families in the orbital region where the group resides.

In the last years, several machine learning algorithms have been introduced in the *Python* programming language and are freely available for solving classification problems. Given a preexisting population, these methods may use the known data to predict if new data belongs, or not, to a given group. In this work we will investigate if these algorithms can be applied for the purpose of automatically identifying new possible members of a given asteroid family, without the need of obtaining a solution for all the other families in the region. We will then verify how the newly identified asteroids compare to those identified by traditional HCM, and we will introduce parameters to quantify the efficiency of the machine learning algorithms. By studying several algorithms that apply standalone or ensemble methods, we aim to identify the method that could perform best for the problem at hand. We will start our work by selecting the asteroid families that are most suited for our analysis, in the next section.

⋆ E-mail: valerio.carruba@unesp.br

2 MACHINE LEARNING CLASSIFICATION OF NEW ASTEROID FAMILIES MEMBERS

In this section, we are going to discuss the implementation of machine learning algorithms for the purpose of classifying new family members. First, we need a consistent way to identify asteroid family members for the training of the algorithms. For this purpose, we turn our attention to the database of asteroid families available at the Asteroid Families Portal (AFP, http://asteroids.matf.bg.ac.rs/fam/properelements.php, Radović et al. (2017), accessed on December 5th, 2019). The web-page based algorithms allow to automatically obtain asteroid families in a data set of 631226 asteroids with synthetic proper elements using standard HCM procedures. New asteroids proper elements, not included in the database used for the family identification purposes, are also available in this site. For each family available in that database, with the exception of the very large Flora and Vesta families, and the complicated cases of the Nysa, Polana, and new Polana families, we divided the family sample into two parts, one for objects with absolute magnitude $H < 14$, the training sample, and one for the rest, which we call the test sample. The choice of this absolute magnitude value as a cutoff is motivated by the work of Milani et al. (2014), that showed that families computed for asteroids with $H < 14$ are not affected by the chaining issue of dynamical families obtained with hierarchical clustering methods. These groups should present minimal or no overlap with other nearby dynamical families.

We required the chosen families to be represented in the Radović et al. (2017) catalog, to be located in the inner, central, and outer main belt at low inclinations (see Carruba et al. (2013) for a definition of these zones), since in these regions we found the most numerous families, and also to have at least 10 members with $H < 14$ (we include just one case with a lower sample, that of the Massalia family, so as to have a larger number of families located in the inner main belt. This family is most likely the outcome of a cratering event, which explains the lack of bright members (Milani et al. 2019)). 21 asteroid families satisfy our selection criteria and will be used for this study. Machine learning methods are applied to the proper $(a, e, sin(i))$ distribution of the training sample, and used to predict the membership of the test sample population. To check how effective the algorithms are in retrieving asteroid families members, we defined three coefficients. f_1, or *"Completeness"* is the fraction of family members that were retrieved by the machine learning algorithm with respect to the total original population. If, following the notation of Carruba et al. (2019), we define as true positive ($TPos$) asteroids identified as family members by both methods, false positive ($FPos$) as asteroids identified as family members by the machine-learning algorithm alone, and false negative ($FNeg$) as asteroids not identified as family members only by the machine learning algorithms, it then follows that:

$$f_1 = Completeness = \frac{TPos}{TPos + FNeg} = \frac{TPos}{N_{Or}}, \quad (1)$$

where $N_{Or} = TPos + FNeg$ is the number of asteroids in the original family. We can also define a *Precision* coefficient that yields the ability of the model to avoid predicting false data. This is given by (see equation (3) in Carruba et al. (2019)):

$$Precision = \frac{TPos}{TPos + FPos} = \frac{TP}{N_{Retr}}, \quad (2)$$

where $N_{Retr} = TPos + FPos$ is the number of asteroids in the retrieved family. A high value of f_1 alone indicates that the algorithm may have been successful at retrieving a large part of the original population. But, this, alone, is not an indication of a good fit. For instance, imagine having an original small family of $\simeq 100$ members and a retrieved family of $\simeq 10000$ objects that includes all 100 original members. The *Completeness* f_1 coefficient would be **1** in this case. But, clearly, the retrieved family is not a good approximation of the original one, in this case. Conversely, a high value of *Precision* alone may be associated with a family that is too small. An example of this could be an original family of $\simeq 100$ members and a retrieved family of $\simeq 10$ objects, all true positives. The *Precision* coefficient would be 100%, but, again, the retrieved family would not be a good representation of the original group. Both coefficients yield useful information, but neither is sufficient alone to identify good retrieved families. As a compromise, and also to use a single parameter instead of two, we introduce the "final parameter" (FP) as:

$$FP = \frac{1}{\sqrt{2}}\sqrt{(Completeness)^2 + (Precision)^2}. \quad (3)$$

In this work we will apply three classes of machine learning algorithms: Standalone, bagging, and boosting methods. Standalone methods, as their name suggests, are approaches that use a single algorithm for the classification process. Bagging and boosting methods are ensemble methods that use several Standalone algorithms. Bootstrap aggregating, often abbreviated as bagging, has each classifier in the ensemble accounted for with an equal weight. An example of bagging methods is the random forest algorithm, where the outcomes of several random decision trees are combined to achieve very high classification accuracy. Boosting does not account for each classifier with a equal weight, but emphasizes the training cases that previous classifiers miss-classified, and the standalone classifiers that performed better. Boosting may provide better accuracy than bagging, but it may also be more likely to over-fit the training data. More details on the theory behind these methods can be found in Swamynathan (2017).

Each algorithm may depend on one or more free parameters, which need to be optimized. For instance, in the k-Nearest Neighbors a free parameter is the number of neighbors to each point. In an ensemble method such as the random forest, the number of single Standalone estimators, or, number of estimators, is another free variable. These parameters, also called hyper-parameters, need to be studied on a case by case basis for each given family. In this work, we will use the *GridSearchCV* approach of the scikit-learn package (Pedregosa et al. 2011), which applies the algorithm of interest for a full grid of values of the hyper-parameter. We will then apply the best-fit value found by this approach to each studied family. Mean values of the hyper-parameters and of the FP coefficients for the 21 studied asteroid families, with their errors, will be provided for each of the studied algo-

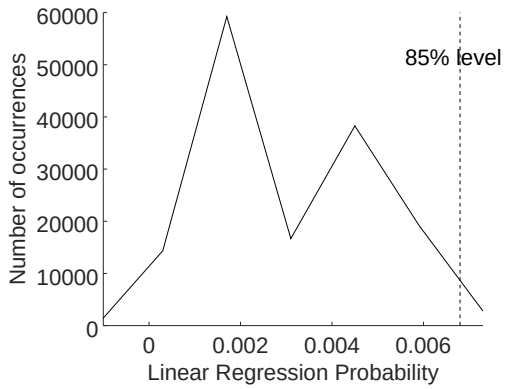

Figure 1. The distribution of probability levels for asteroids in the low-inclination central main belt to belong to the 729 Watsonia family, as obtained by the linear regression algorithm. The vertical dashed line displays a probability value that is higher of those of 85% of the tested population.

rithms. For families not included in our test sample, a given algorithm can then be applied using the mean value of the hyper-parameter found with the analysis of the 21 studied cases. The mean value of the FP coefficient can be used to compare the outcome of different algorithms: the higher its value, the better the algorithm at retrieving family members.

In the next subsection, we will investigate how the selected methods perform the task of classifying new possible members, starting with Standalone algorithms. All tables referring to results obtained in the next subsections are available in the appendix.

2.1 Standalone methods

Here we will investigate the effectiveness of three Standalone machine learning algorithms for the identification of new family members: Linear regression, k-Nearest Neighbors, and Decision tree. We start our analysis with the Linear regression method.

2.1.1 Linear regression algorithm

Linear regression searches for a linear relationship between an independent variable, x or input, and a dependent one, y or output. For our application, the linear regression algorithm uses the proper element data from the family to train, and then computes a probability for asteroids in the extended database that they may belong to the family. The asteroid is assumed to be a family member if its computed probability is higher than a threshold value. Since the relationship between the $(a, e, \sin(i))$ proper elements and family membership may be non-linear, we do not expect this algorithm to perform well. Nevertheless, for historical reasons, and since this is one of the most commonly used machine-learning algorithms, we will begin our analysis by using linear regression.

Figure (1) displays a plot of the probability levels of a body to belong to the (729) Watsonia family, for asteroids in the central main belt. On the y-axis, we report the number of asteroids in the region that have a given probability of belonging to this family. The value of probabilities changes, but we can define them as a percentage of the maximum probability level. The vertical dashed line in figure (1) shows the 85% probability level, which is a probability value that is higher than those computed by the algorithm for 85% of the tested population. Other choices of the probability cut-off are, of course, possible. One has, therefore, to choose the probability level that is most appropriate for a given family. Here we choose the probability level that maximizes values of FP, as computed by equation (3). For each family, we tested various values of the probability cutoff, compute the corresponding *Completeness*, *Precision* and FP coefficients, and selected the probability cutoff associated with the highest FP. Our results for 21 asteroid families are shown in table (A1) in the appendix. The last column displays the values of the hyper-parameter probability cutoff coefficient that satisfy our selection criteria.

For illustrative purposes, we show, for the case of the linear regression algorithm, histograms of hyper-parameter value (the left panel of figure (2) displays a histogram of the "probability cutoff" hyper-parameter), and of the FP coefficient (right panel of figure (2)). Similar histograms were generated for the other algorithms studied in this paper, but will not be shown for the sake of brevity. The mean value of the hyper-parameter was 50.5 ± 11.17, while for FP we obtained $FP = 0.71 \pm 0.02$. Since results of this approach, even at the optimal value of the hyper-parameter, all tend to overestimate the family and yield low values of the *Precision* coefficient, the applications of this method may be limited.

2.1.2 k-Nearest Neighbors (KNN) algorithm

The basic idea is to predict the status of a data point, which, in our context, means if a given asteroid is a member of the family or not, by looking at the k closest neighbors, and then decide by taking a majority vote. For instance, imagine that an asteroid has 5 neighbors, 3 of which are members of the family of interest. In this case, since the majority of the neighbors are family members, the asteroid would also be labeled as such. A hyper-parameter of this method is the optimal "number of neighbors" to be considered. The mean value of this parameter for the 21 studied families was 2.35 ± 1.88, while for FP we obtained $FP = 0.84 \pm 0.08$. Please see table (A2) in the appendix for a summary of our results.

2.1.3 Decision tree algorithm

Decision tree classifications algorithms were first introduced by Quinlan (1986). The concept behind this method is relatively simple: decisions are taken in forms of a tree. Imagine that you want to know the weather of a day in the past. At the top level, we may ask whether the day was sunny or rainy. At a second level, we could ask if the temperature was hot, defined as higher than a threshold, or cold. We could then ask the data if the day was dry or humid, etc. At the final level, the leaf node, the last leaf will classify the data into various categories, like a sunny, warm, dry day, etc. The number of decision nodes to be used in the evaluation of the data, or "max depth", in the implementation of the Decision tree algorithm by scikit-learn, is a hyper-parameter

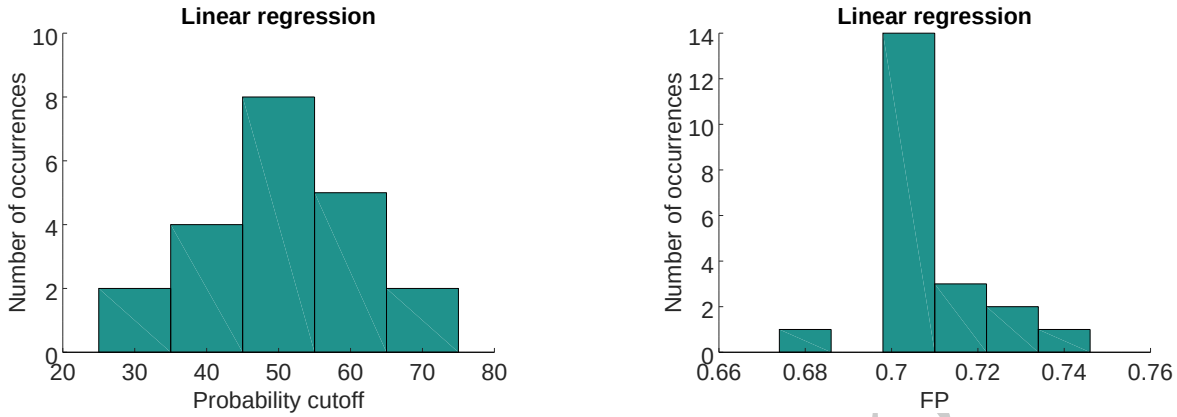

Figure 2. Histograms of the hyper-parameter probability cutoff (left panel) and FP coefficient (right panel) for the 21 families studied with the linear regression algorithm.

of this method. For the studied families, the mean value of this parameter was 7.25 ± 2.91, while for FP we obtained $FP = 0.84 \pm 0.07$. Table (A3) in the appendix reports our results.

2.2 Bagging classifiers

In this subsection we will investigate three cases of bootstrap aggregations: the bagging classifier, the random forest, and the extremely randomized trees.

2.2.1 Bagging classifier

This method, also known as bootstrap aggregation, was first introduced by Leo Breiman in 1994. In this approach, the training data is first used to create multiple samples, called the bootstrap samples. Typically, the bootstrap sample size is the same as the original training sample size. However, it contains 3/4 of the original values and randomly chosen replacements for the remaining 1/4, which may lead to repetition of data. Each of the bootstrap samples is used to train an independent classifier, which can be a linear regression or a decision tree algorithm. For regression problems, where the predicted variable is a continuous one, the final model is built based on the average of the predictions of all standalone classifiers. For classification problems, like the one at hand in this work, where two outcomes are possible (the asteroid is a member of a family, or it is not), the criterion used for determining the final outcome is a majority vote among the standalone classifiers.

Here we use the bagging classifier method using Decision trees as standalone models. The "number of neighbors" hyper-parameter for the Decision trees algorithms is the optimal value found in section (2.1.3). A hyper-parameter of this approach is the "number of estimators", which is the number of standalone algorithms used by the bagging classifier. Here we investigate the values of these hyper-parameter in the range from 1 to 150. The mean value of the "number of estimators" was $15.55^{+18.16}_{-15.55}$, while for FP we obtained $FP = 0.81 \pm 0.09$. Results for the 21 studied families are displayed in table (A4) in the appendix.

2.2.2 Random Forest algorithms

Random forest algorithms (RF) are an application of bagging classifiers methods, with improvements for what concerns the procedure by which the standalone method, which is the Decision tree, selects the optimal split point for the various nodes. In normal applications of Bagging classifiers the decision tree algorithms is allowed to look at all variable values in order to produce the optimal split point for the tree structure. As a consequence, the Decision trees can have a lot of similarities among them, and their predictions may be somewhat correlated. In Random forest algorithms the learning algorithm will only test random samples of variable values. Classification problems are handled by Random forest algorithms in the same way as Bagging classifiers, i.e., by a majority vote of the predictions of each standalone method. A hyper-parameter of this method is also the "number of estimators", which is the number of single decision trees used by the ensemble approach. As in previous sections, we used the *GridSearchCV* approach to find the best values of this parameter in a range from 1 to 150. Our results are shown in table (A5) in the appendix. The mean value of the "number of estimators" hyper-parameter was 40.7 ± 9.56, while for FP we obtained $FP = 0.77 \pm 0.08$.

2.2.3 Extremely Randomized Trees (ExtraTree)

As with the two previous algorithms, this is also a bagging method in which several predictions from different Decision trees are combined to obtain a better final result. With respect to the two previous methods, however, the Extremely Randomized Trees, or ExtraTree, does not use bootstraps, which means that it samples data without replacements. The procedure for creating the tree structure is similar to that of the Random forests algorithm (see discussion in section (2.2.2)). As for other bagging ensemble methods, a hyper-parameter of this algorithm is the "number of estimators". Again, we used the *GridSearchCV* approach to find the best value of this parameter in a range from 1 to 150. Our results are provided in table (A6) in the appendix. The mean value of the "number of estimators" hyper-parameter was 40.95 ± 29.21, while for FP we obtained $FP = 0.84 \pm 0.10$.

2.3 Boosting

Three of the most commonly used boosting algorithms, Adaptive boosting (AdaBoost), Gradient boosting (Gboost), and the eXtreme Gradient boosting (XGboost) will be used in the next sub-subsections.

2.3.1 Adaptive boosting (AdaBoost) algorithm

Boosting methods are, like bagging ones, ensemble methods where the predictions of several standalone approaches are combined to offer a better outcome. With respect to bagging methods, boosting algorithms track the classifiers that provided the least accurate prediction and assign to them lower weights. The final outcome is reached as a weighted average of the outcomes of each standalone classifier. The method used for Adaptive boosting (AdaBoost) follows these steps, interested readers can found further details in Swamynathan (2017), chapter 4. At first, all data points and model receive equal weights. The classifiers are trained over the data points, and the data points that were wrongly classified receive a higher weight, so that in the next iteration they will influence the outcome of the model more than before. Classifier that had a higher accuracy receive higher weights. The process is iterated until the training data is fitted without significant errors, or the maximum number of estimators is reached. As discussed, the final outcome is achieved by means of weighted averages. An important hyper-parameter of this method is the "number of estimators", that has been tuned with *GridSearchCV* to find the optimal value in a range from 1 to 150. The mean value of this hyper-parameter was $41.25^{+45.41}_{-41.25}$, while for FP we obtained $FP = 0.85 \pm 0.06$. Results of the applications of this method are shown in table (A7) in the appendix.

2.3.2 Gradient boosting (Gboost) algorithm

While the AdaBoost algorithm used weights to identify weak standalone algorithms, Gradient boosting (or Gboost) uses gradients. This works in the following way: imagine that one of the standalone classifiers used under-performs when compared to others and needs to be corrected by adding a new estimator. At the next iteration of the process, the new estimator will try to fit the residual difference between the data and the predictions of the former weak learner. The process is then repeated until no further improvements are possible. More detailed information on this approach can be found in Swamynathan (2017), chapter 4. As in section (2.3.1) we use *GridSearchCV* to find the optimal value of the hyper-parameter "number of estimators", which is 91.45 ± 52.67. The mean value of FP was $FP = 0.76 \pm 19$. Results are displayed in table (A8) in the appendix.

2.3.3 eXtreme gradient boosting (XGboost) algorithm

Both the eXtreme gradient boosting (XGboost) and the Gradient boosting (Gboost) methods follow the principle of gradient boosting. XGboost, however, uses a more regularized model, and has performance enhancements, like better support for multi-core processing, that permits for faster training times. An in-depth discussion of the theory behind this method would require much more space, and it is beyond the purposes of this paper. More information can be found in Swamynathan (2017), chapter 4, and references therein. XGBoost depends on a series of hyper-parameters. Among them, here we will work with the following parameters, whose definitions are direct quotes from Swamynathan (2017):

- *eta*: the learning rate, default value is 0.3.
- *max_depth*: Maximum depth of trees, default is 6.
- *colsample_bytree*: The fraction of columns to be randomly sampled for each tree, default value is 1.
- *Subsample*: The fraction of observations to be randomly sampled for each tree algorithm, default is 1.
- *alpha*: L1 regularization term on weight, default is 1.
- *lambda*: L2 regularization term on weights, default value of 1.

To optimize the values of these parameters we used the *GridSearchCV* approach of the scikit-learn package. For each given value of a hyper-parameter, the method fits the data and search for the most optimal values amongst the studied ones. Here we use the following ranges for our hyper-parameters: *eta*: [0.001, 0.01, 0.1], *max_depth*: [2, 5, 10, 20], *colsample_bytree*: [0.1, 0.5, 0.8, 1], *Subsample*: [0.1,0.5,1], *alpha*: [0.1,0.5,1], and *lambda*: [0.1,0.5,1]. As for other ensemble methods, we also tuned the "number of estimators" hyper-parameter, and we tested values in a range from 1 to 150.

Results are shown in table (A9) in the appendix. Values of the *eta* and *Subsample* parameters were equal to 0.1 for all cases. The mean values of the other four parameters and of their errors, estimated to be equal to the standard deviations of the distributions, were *max_depth* = 5±5, *colsample_bytree* = 1.0± 0.1, *alpha* = 0.1*pm* 0.1, and *lambda* =0.5±0.4, respectively. The mean value of the "number of estimators" hyper-parameter was 88.15±46.71, while for FP we obtained $FP = 0.83 \pm 0.09$.

3 THE BIG PICTURE

Each asteroid family is unique in terms of its population, of the effects of the local dynamical environment, its age, its possible interaction with massive asteroids or dwarf planets, etc. It is, therefore, not surprising that the outcome of the studied classification algorithms is different for each of the 21 studied cases. Overall, an important parameter to consider is the size of the sample used to train the classification algorithm. The larger the sample, the better we would expect the outcome of the prediction to be. For each studied family, we verified which algorithm provided the best performance, measured by the FP parameter. Our results are shown in table (1).

Overall, Extra Tree had the best performance for 10 out of the 21 studied families, followed by KNN with 4 cases, XGBoost (3 families), AdaBoost (2 families), and the Bagging Classifier and Decision Tree algorithms, both with 1 case. Extra Tree performed better with families with a large $N(H < 14)$ population, and comes as second best in many other cases (4). KNN appears to perform well for small or medium sized families, with less than $\simeq 150$ members. XGBoost is a third possible good choice among the studied algorithms.

Table 1. Summary of the best results of the studied machine learning algorithms. We report the family identification, the number of members with $H < 14$, the values of the "Completeness" (or f_1), and "Precision" coefficients associated to the best value of the FP coefficient obtained in our study (displayed in the fifth column), and the name of the estimator that outperformed the other algorithms.

Family Id.	Members with $H < 14$	f_1	Prec.	Best FP	Best Estimator
20 Massalia	4	0.35	0.83	0.64	ExtraTree
163 Erigone	16	0.61	0.92	0.78	AdaBoost
15 Eunomia	1072	0.93	0.86	0.89	ExtraTree
170 Maria	392	0.83	0.87	0.85	Bagging Class.
668 Dora	128	0.94	0.98	0.96	ExtraTree
847 Agnia	98	0.76	0.95	0.86	ExtraTree
363 Padua	39	0.81	0.68	0.74	KNN
1726 Hoffmeister	42	0.86	0.94	0.89	KNN
410 Chloris	45	0.80	0.66	0.87	XGBoost
808 Merxia	28	0.85	0.95	0.90	Dec. Tree
128 Nemesis	19	0.70	0.92	0.81	XGBoost
569 Misa	11	0.60	0.98	0.81	AdaBoost
221 Eos	2043	0.88	0.90	0.89	ExtraTree
24 Themis	1029	0.95	0.94	0.95	ExtraTree
158 Koronis	790	0.97	0.95	0.96	ExtraTree
10 Hygiea	453	0.90	0.95	0.93	XGBoost
375 Ursula	285	0.80	0.89	0.85	ExtraTree
1040 Klumpkea	195	0.89	0.94	0.91	ExtraTree
283 Emma	37	0.89	0.89	0.89	KNN
845 Naema	18	0.92	0.95	0.94	ExtraTree
490 Veritas	133	0.96	0.91	0.94	KNN

Table 2. Results of KNN and ExtraTree for five highly inclined ($\sin(i) > 0.3$) asteroid families.

Family Id.	Members with $H < 14$	KNN FP	ExtraTree FP
480 Hansa	347	0.89	0.90
945 Barcelona	88	0.84	0.88
31 Euphrosyne	163	0.91	0.95
702 Alauda	459	0.81	0.86
87 Sylvia	32	0.72	0.74

Figure (3) displays an $(a, sin(i))$ projection of the best results for family membership of the ExtraTree algorithm (Koronis family, panel (a)), and KNN method (Veritas family, panel (b)). Black dots display the members of the family, as obtained by the AFP, while red circles show the orbital locations of the family members predicted by the machine learning algorithm. Values of FP are higher than 94% for both families, with an excellent performance of the two methods.

To check if the two algorithms that produced the best results also perform well with other families, we studied the case of five asteroid families in the Cybele and highly inclined main belt, that have a statistically significant population of objects with $H < 14$: the families of (480) Hansa, (945) Barcelona, (31) Euphrosyne, (702) Alauda, and (87) Sylvia. We use the optimal values of the hyper-parameters found in our previous analysis, which is 2 for the number of neighbors in KNN and 41 for the number of estimators in the ExtraTree algorithm. Our results are shown in table (2).

ExtraTree had the best performance in all cases, with values of FP consistently higher than 74%. Based on this analysis, we believe that this algorithm may generally be the best tool to use amongst the studied methods.

4 UPDATING THE MEMBERSHIP OF ASTEROID FAMILIES

The main goal of a classification algorithm should be to find new family members. Having established what is the optimal tool, we now try to use it to update the list of family members. We modify the approach and divide the sample of proper elements in a given region into two parts. The first part will go up to the highest numbered member in a given asteroid family. Asteroids in this set of proper elements will be labeled with 1 if they belong to a given family, and 0 otherwise. Machine learning algorithms will then use this set of elements to train so as to predict which asteroids in the remaining set of asteroids with identifications larger than that of the highest numbered object in the family are most likely to be classified as new family members.

We applied this new approach to four asteroid families, the (694) Ekard (105 members), (480) Hansa (1484 members), (15) Eunomia (6076 members), and (832) Karin (480 members). These are a small, medium, and large groups in terms of numbers of members. Karin is a sub-family of the larger Koronis group and was included to test how the algorithm performs for sub-families located in high-number density regions. Contrary to the tests that we run before, we quantify the quality of the fit by checking how the retrieved family compares with respect to the known family in terms of the dispersions of proper elements, since, in principle, we have no information on the family obtainable by standard

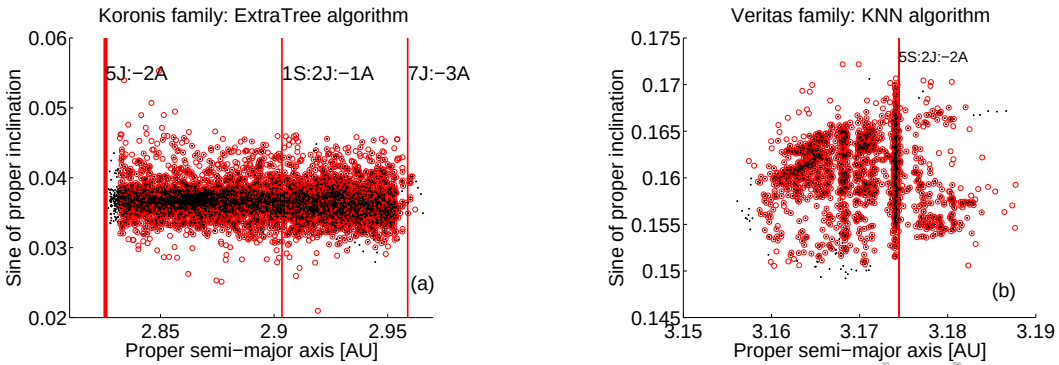

Figure 3. An $(a, \sin(i))$ projection of members of the sampled (black dots) and predicted (red circles) Koronis (panel a) and Veritas (panel (b) asteroid families. Members of the Koronis family were predicted using the ExtraTree algorithm, while members of the Veritas family were determined with the KNN method. Vertical lines display the location of local two- and three-body mean-motion resonances.

HCM in the extended sample. In particular, we define an index d_{std}:

$$d_{std} = \frac{1}{\sqrt{3}}\sqrt{\left(\frac{\sigma(a_{retrieved})}{\sigma(a_{known})}\right)^2 + \left(\frac{\sigma(e_{retrieved})}{\sigma(e_{known})}\right)^2 + \left(\frac{\sigma(\sin(i)_{retrieved})}{\sigma(\sin(i)_{known})}\right)^2} \quad (4)$$

where σ is associated with the standard deviation of the distribution of $(a, e, \sin(i))$ proper elements, and the retrieved and known subscripts refer to the extended family obtained with the algorithm and the known family used for the fitting process. A d_{std} close to 1 implies that the distribution in proper elements of members of the extended family is compatible with the distribution of the retrieved family, and that results of the machine learning algorithm may be reasonable.

The first three panels of figure (4) display $(a, \sin(i))$ projections of three studied families, while the fourth shows a (a, e) section of the fourth, the Karin sub-family of the Koronis group. Black dots identify the original population, while red circles show the new possible family members identified by the ExtraTree algorithm. The scatter in the third dimension for these groups is similar to that observed in the two-dimensional orbital sections. While the distribution of proper elements is tighter in the proper $(a, \sin(i))$ plane for the Ekard, Hansa, Eunomia, and, generally speaking, for most asteroid families, for the very young Karin family the distribution in proper (a, e) still mostly reflects the dispersion caused by the initial ejection velocity field, which is why for this family we decided to plot proper elements in this domain.

We identify 37 new members for the Ekard family, 694 for Hansa, 1550 for Eunomia, and 324 for the Karin group. In all cases, the orbital distribution of the new members is quite compatible with that of the old ones: overall, the mean value of d_{std} was of 0.97 ± 0.04, where the error is defined as the standard deviation of d_{std}. Results are similar for other asteroid families, and the algorithm is able to accurately identify even members of sub-families located in dense regions of the main belt, such as Karin. Overall, ExtraTree appears to be a reliable tool for automatically detecting new possible family members.

5 CONCLUSIONS

In this work, we investigated the possibility of using machine learning classification algorithms for identifying new asteroid families members. We selected 21 asteroid families that, with the exception of the family of (20) Massalia, have a statistically significant population of objects with $H < 14$, which is less likely to be affected by the chaining issues that affect standard HCM. We then applied nine different machine learning algorithms and used three parameters to characterize the efficiency with which the algorithms identify new family members when compared to the results of traditional HCM. We used Standalone, bagging classifiers, and boosting methods for this purpose. Among the studied algorithms, the Extremely Randomized Trees (ExtraTree) had the best performance, followed by the k-Nearest Neighbors (KNN) Standalone approach and by the eXtreme gradient boosting (XGBoost) method. An application to highly inclined asteroid families confirmed that ExtraTree appears to be the most efficient method for this particular classification task.

We adapted the codes for the purpose of automatically updating family memberships, and we tested them with the cases of small, medium-sized and large asteroid families. In all tested cases, the orbital distribution of the newly found members is highly consistent with the distribution of previously known objects. The *Python* codes used for this task are publicly available at the GitHub repository (https://github.com/valeriocarruba/Machine-learning-classification-of-new-asteroid-families-members, information on how to download and use the codes is available in the local read-me files.) We welcome inputs from the scientific community on how to further improve the performance of these algorithms.

6 APPENDIX

In this appendix, we report the results of the application of nine machine learning algorithms to 21 asteroid families. See section 2 for the descriptions of the algorithms and of the listed parameters.

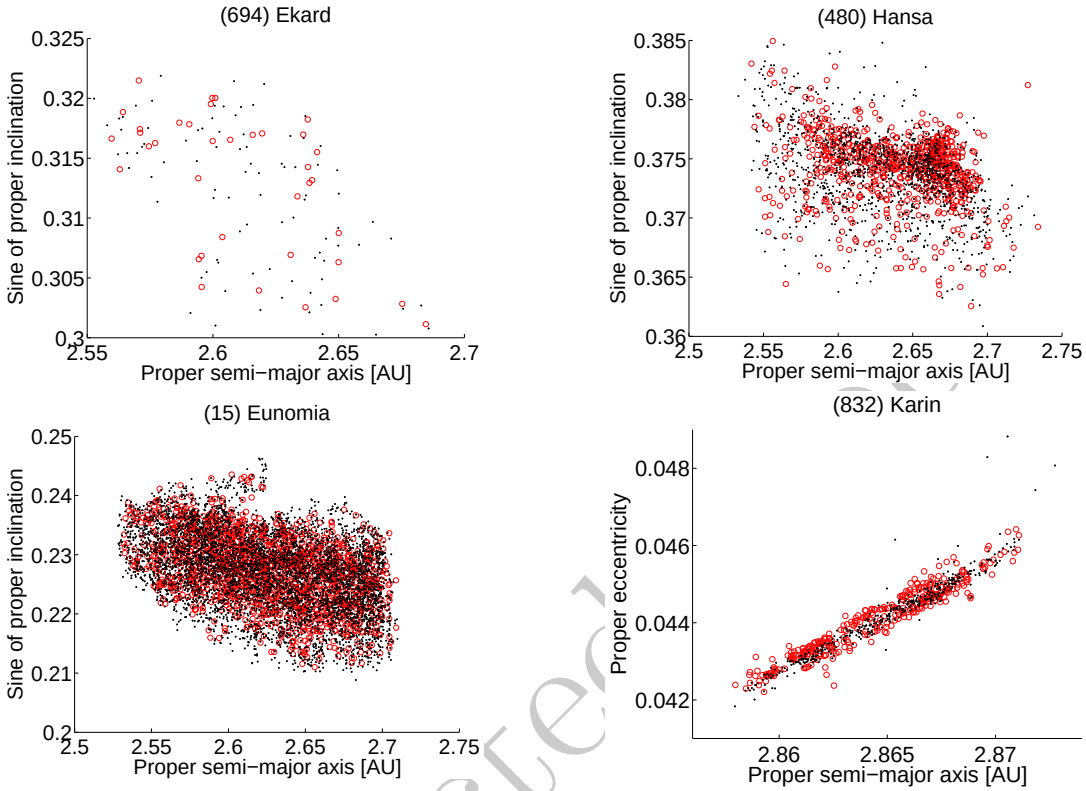

Figure 4. Proper $(a, \sin(i))$ projections of known members (black dots) and retrieved members (red circles) of the Ekard, Hansa, and Eunomia asteroid families. The fourth panel displays an (a, e) projection of members of the known and retrieved Karin sub-family.

Table A1. The table reports the asteroid families from the Asteroid families Portal, the number of their members, the number of members with $H < 14$, the optimal values of "Completeness" (f_1), "Precision" ($Prec$), and "final parameter" FP coefficients, and the corresponding value of the hyper-parameter probability cutoff, for the families studied with the linear regression algorithm.

Family Id.	Number of members	Number of members $H < 14$	f_1	$Prec$	FP	Cutoff
20 Massalia	6400	4	1.00	0.14	0.71	50.0
163 Erigone	2979	16	0.96	0.09	0.68	30.0
15 Eunomia	6076	1072	1.00	0.10	0.72	40.0
170 Maria	4183	392	1.00	0.12	0.71	40.0
668 Dora	1677	128	1.00	0.04	0.71	50.0
847 Agnia	4432	98	1.00	0.12	0.71	60.0
363 Padua	807	39	1.00	0.02	0.71	60.0
1726 Hoffmeister	2396	42	0.99	0.09	0.70	60.0
410 Chloris	449	45	1.00	0.01	0.71	30.0
808 Merxia	1624	28	1.00	0.03	0.71	50.0
128 Nemesis	1449	19	1.00	0.02	0.71	50.0
569 Misa	723	11	1.00	0.03	0.71	60.0
221 Eos	14661	2043	1.00	0.19	0.72	40.0
24 Themis	5946	1029	1.00	0.29	0.74	50.0
158 Koronis	7294	790	0.93	0.47	0.73	60.0
10 Hygiea	6224	453	1.00	0.12	0.71	40.0
375 Ursula	2214	285	0.99	0.21	0.72	70.0
1040 Klumpkea	2840	195	0.99	0.25	0.73	50.0
283 Emma	610	37	1.00	0.01	0.71	50.0
845 Naema	418	18	1.00	0.08	0.71	70.0
490 Veritas	1805	133	0.99	0.11	0.71	50.0

Table A2. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "number of neighbors" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$ and FP coefficients for families identified with the KNN algorithm.

Family Id.	Number of neighbors	f_1	Prec	FP
20 Massalia	1	0.39	0.80	0.63
163 Erigone	1	0.70	0.76	0.73
15 Eunomia	4	0.89	0.85	0.87
170 Maria	4	0.81	0.85	0.83
668 Dora	2	0.93	0.96	0.95
847 Agnia	1	0.87	0.82	0.85
363 Padua	1	0.81	0.68	0.74
1726 Hoffmeister	1	0.86	0.94	0.89
410 Chloris	2	0.77	0.67	0.72
808 Merxia	4	0.56	0.91	0.75
128 Nemesis	2	0.57	0.92	0.76
569 Misa	1	0.68	0.74	0.71
221 Eos	1	0.89	0.85	0.87
24 Themis	1	0.94	0.91	0.93
158 Koronis	1	0.96	0.92	0.94
10 Hygiea	2	0.80	0.95	0.88
375 Ursula	1	0.84	0.80	0.82
1040 Klumpkea	8	0.91	0.90	0.90
283 Emma	6	0.89	0.89	0.89
845 Naema	2	0.99	0.81	0.91
490 Veritas	2	0.96	0.91	0.94

Table A3. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "Max Depth" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$ and FP coefficients for families identified with the Decision tree algorithm.

Family Id.	Max of Depth	f_1	Prec	FP
20 Massalia	3	0.22	0.81	0.60
163 Erigone	7	0.63	0.89	0.77
15 Eunomia	12	0.90	0.81	0.86
170 Maria	6	0.85	0.81	0.83
668 Dora	6	0.78	0.92	0.85
847 Agnia	6	0.86	0.80	0.83
363 Padua	4	0.78	0.68	0.73
1726 Hoffmeister	5	0.66	0.98	0.83
410 Chloris	6	0.77	0.65	0.71
808 Merxia	5	0.85	0.95	0.90
128 Nemesis	5	0.73	0.89	0.81
569 Misa	4	0.73	0.62	0.68
221 Eos	14	0.87	0.84	0.85
24 Themis	7	0.94	0.91	0.93
158 Koronis	9	0.96	0.96	0.96
10 Hygiea	9	0.89	0.92	0.91
375 Ursula	14	0.74	0.79	0.76
1040 Klumpkea	7	0.85	0.90	0.88
283 Emma	8	0.79	0.92	0.85
845 Naema	5	0.80	0.97	0.89
490 Veritas	6	0.87	0.94	0.91

ACKNOWLEDGMENTS

We are grateful to an anonymous reviewer for comments and suggestions that helped improve this article. We would like to thank the São Paulo State Science Foundation (FAPESP, grant 2018/20999-6), the Brazilian National Research Council (CNPq, grant 301577/2017-0, 310317/2016-9), and the Coordination for the Improvement of Higher Education Personnel (CAPES, grant 88887.374148/2019-00). We acknowledge the use of data from the Asteroid Dynamics Site (AstDys)

Table A4. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "number of estimators" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$, and FP coefficients for families identified with the "bagging classifier" algorithm.

Family Id.	Number of estimators	f_1	Prec	FP
20 Massalia	1	0.01	0.67	0.47
163 Erigone	4	0.31	0.92	0.69
15 Eunomia	32	0.90	0.74	0.82
170 Maria	19	0.83	0.87	0.85
668 Dora	19	0.72	0.94	0.84
847 Agnia	14	0.84	0.89	0.86
363 Padua	2	0.51	0.80	0.67
1726 Hoffmeister	5	0.53	0.94	0.76
410 Chloris	4	0.73	0.89	0.81
808 Merxia	6	0.70	0.87	0.79
128 Nemesis	5	0.46	0.63	0.55
569 Misa	3	0.77	0.64	0.71
221 Eos	4	0.90	0.74	0.83
24 Themis	80	0.87	0.84	0.86
158 Koronis	17	0.96	0.95	0.96
10 Hygiea	2	0.89	0.93	0.91
375 Ursula	42	0.88	0.76	0.82
1040 Klumpkea	3	0.88	0.92	0.90
283 Emma	14	0.70	0.95	0.84
845 Naema	14	0.57	0.96	0.79
490 Veritas	22	0.75	0.96	0.86

Table A5. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "Number of estimators" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$ and FP coefficients for families identified with the Random forest method.

Family Id.	Number of neighbors	f_1	Prec	FP
20 Massalia	46	0.00	0.00	0.00
163 Erigone	37	0.07	0.97	0.69
15 Eunomia	48	0.68	0.95	0.83
170 Maria	48	0.60	0.96	0.80
668 Dora	29	0.55	1.00	0.81
847 Agnia	47	0.56	0.99	0.81
363 Padua	46	0.22	0.85	0.62
1726 Hoffmeister	49	0.36	0.99	0.75
410 Chloris	46	0.43	0.94	0.73
808 Merxia	28	0.60	1.00	0.82
128 Nemesis	35	0.23	0.94	0.69
569 Misa	45	0.10	0.92	0.66
221 Eos	40	0.61	0.97	0.81
24 Themis	46	0.82	0.98	0.90
158 Koronis	47	0.91	0.98	0.94
10 Hygiea	49	0.67	0.99	0.85
375 Ursula	49	0.36	0.96	0.72
1040 Klumpkea	16	0.63	0.99	0.83
283 Emma	41	0.22	0.96	0.71
845 Naema	22	0.24	1.00	0.73
490 Veritas	46	0.58	0.98	0.81

(http://hamilton.dm.unipi.it/astdys, Knežević and Milani (2003)) and the Asteroid Families Portal (*AFP*, http://asteroids.matf.bg.ac.rs/fam/properelements.php, Radović et al. (2017)). We also acknowledge the use of exercises proposed by machine learning courses at DataCamp (www.datacamp.com), that either inspired or helped in the development of the software used in this work. Finally, we are grateful to Edmilson Roma de Oliveira for discussions that motivated this work.

Table A6. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "Number of estimators" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$ and FP coefficients for families identified with the ExtraTree method.

Family Id.	Number of estimators	f_1	Prec	FP
20 Massalia	1	0.35	0.83	0.64
163 Erigone	21	0.22	0.90	0.66
15 Eunomia	80	0.93	0.86	0.89
170 Maria	9	0.80	0.86	0.83
668 Dora	85	0.94	0.98	0.96
847 Agnia	67	0.76	0.95	0.86
363 Padua	9	0.52	0.86	0.71
1726 Hoffmeister	25	0.62	0.99	0.83
410 Chloris	43	0.74	0.86	0.80
808 Merxia	62	0.33	0.94	0.70
128 Nemesis	9	0.28	0.85	0.63
569 Misa	5	0.63	0.83	0.73
221 Eos	97	0.88	0.90	0.89
24 Themis	73	0.95	0.94	0.95
158 Koronis	38	0.97	0.95	0.96
10 Hygiea	45	0.85	0.95	0.90
375 Ursula	135	0.80	0.89	0.85
1040 Klumpkea	104	0.89	0.94	0.91
283 Emma	65	0.75	0.96	0.86
845 Naema	5	0.92	0.95	0.94
490 Veritas	42	0.92	0.94	0.93

Table A7. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "Number of estimators" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$ and FP coefficients for families identified with the AdaBoost method.

Family Id.	Number of estimators	f_1	Prec	FP
20 Massalia	2	0.22	0.81	0.60
163 Erigone	147	0.61	0.92	0.78
15 Eunomia	42	0.93	0.81	0.87
170 Maria	62	0.77	0.90	0.84
668 Dora	66	0.84	0.98	0.92
847 Agnia	20	0.73	0.97	0.86
363 Padua	11	0.36	0.93	0.70
1726 Hoffmeister	3	0.57	0.99	0.81
410 Chloris	3	0.82	0.87	0.85
808 Merxia	3	0.61	0.96	0.80
128 Nemesis	4	0.71	0.91	0.81
569 Misa	4	0.60	0.98	0.81
221 Eos	144	0.88	0.87	0.88
24 Themis	92	0.92	0.95	0.94
158 Koronis	8	0.96	0.96	0.96
10 Hygiea	7	0.89	0.93	0.91
375 Ursula	50	0.72	0.84	0.78
1040 Klumpkea	78	0.82	0.93	0.88
283 Emma	72	0.67	0.92	0.81
845 Naema	5	0.64	0.97	0.82
490 Veritas	4	0.87	0.94	0.91

REFERENCES

Carruba V., Domingos R. C., Nesvorný D., Roig F., Huaman M. E., Souami D., 2013, MNRAS, 433, 2075

Carruba V., Aljbaae S., Lucchini, A., 2019, MNRAS, 488, 1377

DeMeo F. E., Alexander C. M. O'D., Walsh K. J., Chapman C. R., Binzel R. P., 2015, Asteroids IV, Patrick Michel, Francesca E. DeMeo, and William F. Bottke (eds.), University of Arizona Press, Tucson, 895, 13

Knežević Z., Milani A., 2003, A&A, 403, 1165

Milani A., Cellino A., Knežević Z., Novaković B., Spoto F., Paolicchi P., 2014, Icarus, 239, 46

Milani A., Knežević Z., Spoto F., Paolicchi P., 2019, A&A 622,

Table A8. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "Number of estimators" hyper-parameter used for each of the studied families, and the values of the f_1, $Prec$ and FP coefficients for families identified with the Gradient Boosting method.

Family Id.	Number of estimators	f_1	Prec	FP
20 Massalia	2	0.30	0.80	0.60
163 Erigone	83	0.59	0.89	0.75
15 Eunomia	127	0.90	0.83	0.87
170 Maria	107	0.78	0.86	0.82
668 Dora	133	0.80	0.99	0.90
847 Agnia	68	0.79	0.86	0.82
363 Padua	9	0.70	0.50	0.61
1726 Hoffmeister	7	0.51	0.69	0.60
410 Chloris	47	0.80	0.55	0.69
808 Merxia	42	0.70	0.74	0.72
128 Nemesis	78	0.14	0.12	0.13
569 Misa	12	0.36	0.57	0.48
221 Eos	131	0.85	0.84	0.85
24 Themis	141	0.96	0.95	0.96
158 Koronis	133	0.96	0.95	0.96
10 Hygiea	140	0.88	0.95	0.92
375 Ursula	135	0.79	0.83	0.81
1040 Klumpkea	147	0.88	0.91	0.90
283 Emma	143	0.56	0.93	0.81
845 Naema	1	0.81	0.75	0.78
490 Veritas	145	0.80	0.95	0.88

Table A9. The table reports the asteroid families from the Asteroid families Portal, the optimal value of the "number of estimators", eta, max_depth, colsample_bytree, Subsample, alpha and lambda hyper-parameters used for each of the studied families, and the values of the f_1, Precision ($Prec$) and FP coefficients for families identified with the eXtreme Gradient Boosting (XGBoost) method.

Family Id.	Number of estimators	eta	max_depth	colsample_bytree	Subsample	alpha	lambda	f_1	Prec	FP
20 Massalia	46	0.1	2	1.0	0.1	0.1	0.5	0.18	0.80	0.58
163 Erigone	136	0.1	5	0.8	0.1	0.1	0.1	0.48	0.97	0.77
15 Eunomia	136	0.1	20	1.0	0.1	1.0	0.1	0.90	0.89	0.89
170 Maria	125	0.1	20	1.0	0.1	0.1	1.0	0.80	0.90	0.85
668 Dora	87	0.1	2	0.8	0.1	0.1	0.1	0.82	0.99	0.91
847 Agnia	141	0.1	5	1.0	0.1	0.5	1.0	0.82	0.92	0.87
363 Padua	45	0.1	5	1.0	0.1	0.5	1.0	0.58	0.70	0.64
1726 Hoffmeister	13	0.1	5	1.0	0.1	0.1	0.5	0.52	0.98	0.78
410 Chloris	24	0.1	5	1.0	0.1	0.1	0.1	0.80	0.66	0.73
808 Merxia	61	0.1	5	0.8	0.1	1.0	0.5	0.57	0.96	0.79
128 Nemesis	66	0.1	5	0.8	0.1	0.1	1.0	0.70	0.92	0.82
569 Misa	17	0.1	5	1.0	0.1	0.1	1.0	0.05	0.92	0.65
221 Eos	145	0.1	20	1.0	0.1	0.1	0.1	0.90	0.85	0.87
24 Themis	146	0.1	20	1.0	0.1	0.1	0.5	0.94	0.94	0.94
158 Koronis	91	0.1	5	1.0	0.1	0.1	0.1	0.96	0.97	0.96
10 Hygiea	105	0.1	10	1.0	0.1	0.1	0.1	0.90	0.95	0.93
375 Ursula	123	0.1	10	1.0	0.1	1.0	1.0	0.79	0.87	0.83
1040 Klumpkea	68	0.1	5	1.0	0.1	0.1	0.1	0.76	0.95	0.86
283 Emma	143	0.1	10	0.8	0.1	0.1	0.1	0.65	0.96	0.82
845 Naema	15	0.1	5	1.0	0.1	1.0	1.0	0.81	0.87	0.84
490 Veritas	76	0.1	10	1.0	0.1	0.1	0.5	0.86	0.95	0.91

A47

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V. et al., 2011, JMLR 12, 2825

Quinlan J. R., 1986, Machine Learning 1, 81

Radović V., Novaković B., Carruba V., Marceta D., 2017, MNRAS, 470, 576

Swamynathan M. 2017, Mastering Machine Learning with Python in six steps, ed. Apress

Zappalá V., Cellino A., Farinella, P., Knežević Z., 1990, AJ, 100, 2030

Zappalá V., Bendjoya Ph., Cellino A., Farinella, P., Froeschlé, C. 1995, Icarus, 116, 291

This paper has been typeset from a TEX/LATEX file prepared by the author.

A machine learning algorithm based on decision trees is called Random Forest Trees (RFT). Machine learning algorithms that do ensemble classification include Random Trees (RT). The term "ensemble" denotes a technique that averages the forecasts of various different base models to provide predictions.
The core idea behind ensemble methods based on randomization is to "incorporate random perturbations into the learning procedure to build multiple alternative models from a single learning set L and then to aggregate the predictions of those models to make the ensemble prediction" (Louppe, 2014). In other words, "growing an ensemble of trees and letting them vote for the most popular class has resulted in significant gains in classification accuracy. These ensembles are frequently grown by creating random vectors that control how each tree in the ensemble grows (Breiman, 2001).

When building a random tree, there are three basic options available. These three considerations are: (1) how to separate leaves; (2) what kind of predictor to utilize in each leaf; and (3) how to introduce unpredictability into trees (Denil et al., 2014). Using a bootstrapped or sub-sampled data set to generate each tree is a typical method for adding unpredictability to a tree. As a result, there are variances among the trees in the forest since each tree in the forest was trained using slightly different data (Denil et al., 2014). The optimal split at a particular node can alternatively be chosen randomly; tests have shown, however, that where noise is relevant, bagging typically produces better results (Louppe, 2014).

"Special attention must be taken so that the resulting model is neither too simple nor too complex," according to the author, when optimizing a Random Trees model. The model is in fact stated to have underfitted the data in the first scenario, i.e., it was not adaptable enough to capture the structure between X and Y. The model is said to be overfit the data in the latter scenario because it is too flexible and captures isolated structures (i.e., noise) that are unique to the learning set (Louppe, 2014).

In order to prevent overfitting, stropping rules must be established to stop a tree from developing before it has too many levels: User-defined hyper-parameters are used to establish stopping conditions (Louppe, 2014). The most popular of these parameters are:
The bare minimum of samples that a terminal node needs to divide
the bare minimum of samples in a leaf node after splitting the terminal node
The maximum depth of a tree, or the number of levels it can reach,
once the Gini Impurity index, which measures the Trees accuracy, falls below a predetermined threshold
To identify the best trade-off, these parameters must be fine-tuned; they must be neither too stringent nor too loose for the tree to be neither too shallow nor too deep (Louppe, 2014).
Breiman (2002) lists the following as some of the essential characteristics of random trees:
It is a very good classifier, with accuracy on par with support vector machines.
As the forest grows, it produces an internal, unbiased estimate of the generalization error.
When up to 80% of the data are missing, it nevertheless retains accuracy thanks to an efficient estimation algorithm.
It has a technique for balancing inaccuracy in data sets with an imbalanced class population.

The generated forests can be saved for use on other data in the future.
It provides an estimate of the variables that are crucial for classification.
Information regarding the relationship between the variables and the categorization is shown in the output that is produced.
It calculates distances between examples that can be used for grouping, finding outliers, or scaling to provide intriguing data visualizations.
Contrary to the Support Vector Machine (SVM), the random trees classifier can typically handle a mix of categorical and numerical variables. As for data scaling, Random Trees are less susceptible to it than SVM, which frequently requires data to be normalized before training or classification. SVM is said to perform better, nonetheless, when the training set is little or uneven. Comparable in computational complexity to SVM, the Random Trees classifier performs better and more quickly with big training sets.